NASA Technical Memorandum 105157

Description of Real-Time Ada Software Implementation of a Power System Monitor for the Space Station Freedom PMAD DC Testbed

Kimberly Ludwig
Sverdrup Technology, Inc.
Lewis Research Center Group
Brook Park, Ohio

Michael Mackin and Theodore Wright
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio

Prepared for the
26th Intersociety Energy Conversion Engineering Conference
cosponsored by the ANS, SAE, ACS, AIAA, ASME, IEEE, and AIChE
Boston, Massachusetts, August 4–9, 1991

NASA

IMPLEMENTATION OF A POWER
STATION
(NASA) CSCL
Unclas

Description of Real-Time Ada Software Implementation of a Power System Monitor for the Space Station Freedom PMAD DC Testbed

Kimberly Ludwig
Sverdrup Technology, Inc.
Lewis Research Center Group
Brook Park, Ohio 44142

Michael Mackin and Theodore Wright
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

Abstract

This paper describes the Ada language software developed to perform the electrical power system monitoring functions for the NASA Lewis Research Center's Power Management and Distribution (PMAD) DC testbed. The results of the effort to implement this monitor will be presented. The PMAD DC testbed is a reduced-scale prototype of the electric power system to be used in Space Station Freedom. The power is controlled by smart switches known as power control components (or switchgear). The power control components are currently coordinated by five Compaq 386/20e computers connected through an 802.4 local area network. One of these computers is designated as the control node with the other four acting as subsidiary controllers. The subsidiary controllers are connected to the power control components with a Mil-Std-1553 network. An operator interface is supplied by adding a sixth computer.

The power system monitor algorithm is comprised of several functions including: periodic data acquisition, data smoothing, system performance analysis, and status reporting. Data is collected from the switchgear sensors every 100 milliseconds, then passed through a 2 Hz digital filter. System performance analysis includes power interruption and overcurrent detection. The reporting mechanism notifies an operator of any abnormalities in the system. Once per second, the system monitor provides data to the control node for further processing, such as state estimation.

The system monitor required a hardware timer interrupt to activate the data acquisition function. The execution time of the code was optimized by using an assembly language routine. The routine allows direct vectoring of the processor to Ada language procedures that perform periodic control activities. A summary of the advantages and side effects of this technique will be discussed

Introduction

The NASA Power Management and Distribution testbed is designed to evaluate electric power control hardware and software for use in Space Station Freedom, as described in [1]. Presently, the testbed's power control components are controlled by five Compaq 386/20e computers. One computer is designated as the control node, which coordinates each of the subsidiary controllers and provides an interface to the operator. This computer is called the Power Management Controller, or PMC. There is a Photovoltaic Controller (PVC), whose functions are to control and monitor the Sequential Shunt Unit or Solar Array Switching Unit, the DC Switching Unit, and the Battery Charge/Discharge Units. The main feeder lines, crossties, and DC-to-DC converter units are controlled and monitored by the Main Bus Controller (MBC). The last two subsidiary controllers, the Secondary Power Controller (SPC) and Tertiary Power Controller (TPC), are used to communicate to the Remote Bus Isolators, Remote Power Controllers, and the Load Converters. The computers are connected to one another through an 802.4 local area network. Each of the subsidiary computers are connected to power control components with a Mil-Std-1553 bus. The single power channel configuration, shown from a controls viewpoint, of the PMAD testbed hardware is shown in Figure 1.

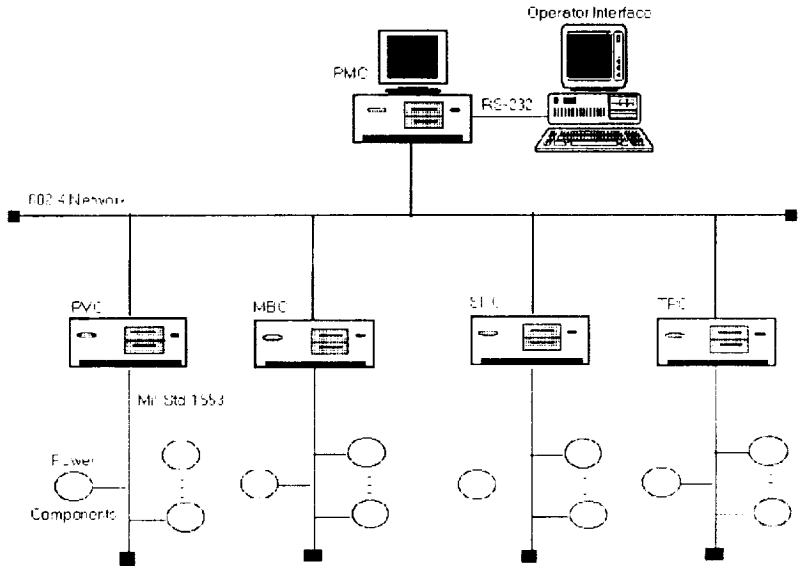

Figure 1 - PMAD Testbed Hardware.

In the next phase of the testbed, this basic configuration is expanded to include a second power channel. The subsidiary controllers are duplicated to handle the additional power sources and loads. An additional difference between the two configurations is that an intermediate controller is inserted between the PMC and the SPCs and TPCs. This controller is the Load Management Controller (LMC) whose function is to coordinate the SPCs and TPCs on each channel. The configuration of the second phase of the testbed hardware is illustrated in Figure 2.

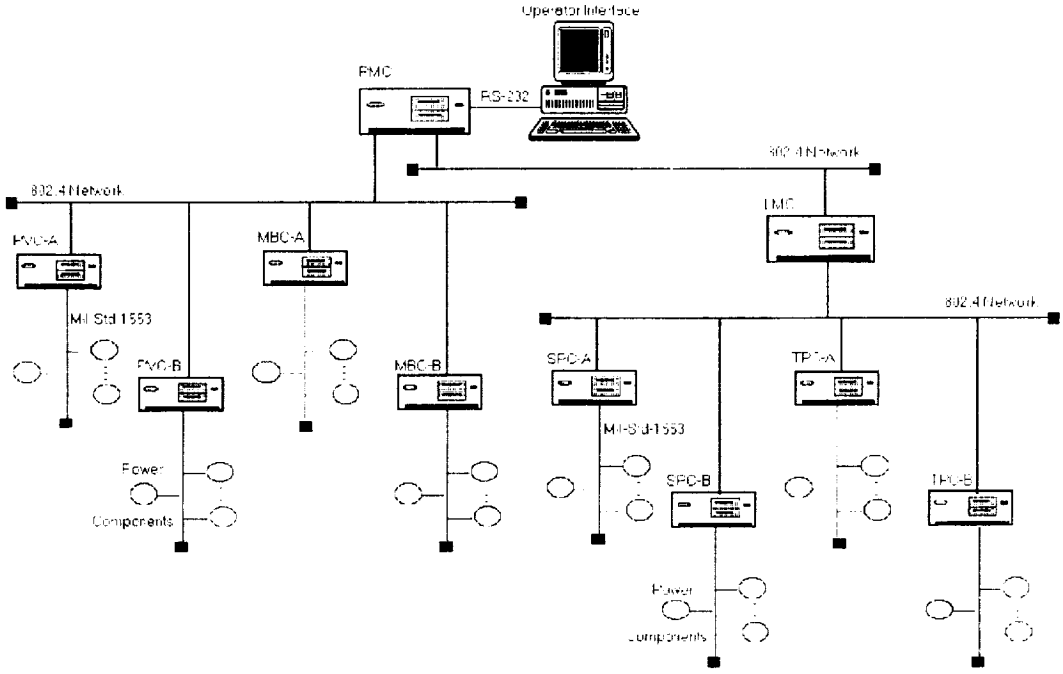

Figure 2 -Dual channel PMAD Testbed Hardware.

The Ada programming language was selected to provide a flexible and maintainable environment for the development of power system control algorithms. The software design for the control

computers uses an object oriented design methodology and incorporates some of the advanced features of the Ada language, including tasking. The software is designed to be as generic as possible, allowing it to be used on any of the five control computers. Hardware dependent portions of the design are encapsulated so that the software can be easily adapted to new power system hardware under development. As these new hardware components are added to the PMAD testbed, new component interfaces will be incorporated into the software. Additionally, new control algorithms will be added as they are developed. Further details on the Ada software design and capabilities can be found in [2].

Software System Design

Since the testbed software is object based, a software "*object*" is associated with every network, power control component, and power control algorithm in the testbed. In Ada, these objects are represented as tasks. Some objects are comprised of two tasks, an "*actor*" and a "*listener*" task. The purpose of the listener task is to provide an interface to other objects in the system. It accepts and processes commands from the other objects. The listener task also has the capability to create, stop, or abort its related actor task. The actor task executes independently and performs any computations that must be done continuously. An example of this type of object is the System_Monitor, described later. Other objects, such as the power control components, have only one task identified with them: a listener task. The interface provided by the listener task is used to request data from, or send commands to the switchgear.

To facilitate communication between the objects in the system, a messaging concept was developed. The *router* is an object that was developed to pass messages. Whenever any object wishes to talk to another object, it "sends a message" to the other object by employing calls to router functions. Additionally, the router provides services to start new control algorithms and power component tasks, assign an alias to an existing task, and display a list of the tasks started on the local processor.

Besides the router object, there are a few other basic objects in the system. The Standard_In object accepts keyboard input, while Standard_Out displays information on the screen. Similarly the Remote_In object accepts input from the serial port, and Remote_Out sends output to the serial port. The Text_Interface task formats data that is to be shown on a monitor. There is a Network object, whose purpose is to provide communications over the 802.4 local area network. All of these tasks are started when the testbed software is invoked.

All power control component and algorithms objects are created at run time. The operator will either type in commands to start these tasks, or read from a file to automatically start them. Defined within every power control component object is a procedure that must be executed in a periodic manner in order for the power system monitor algorithm to execute correctly. This procedure is always named Sync_Proc. Since new component objects can be started at any time, the system monitor algorithm needs to be notified of their existence before starting to process the synchronous[1] operations.

Power System Monitor Algorithm Description

The power system monitor algorithm is the process of periodic data collection, data smoothing, performance analysis, and error reporting. A key operation in this process is periodic sensor data collection. The sampling period of the sensor data is 100 milliseconds. After being acquired in a binary format, the raw data must be converted to decimal numbers. Then it is passed through a digital low pass filter for smoothing. Currently, a 2 Hz third-order Butterworth filter is being used for data filtering. The filtered data is then analyzed for system abnormalities. The types of faults that are detected consist of power interruption, undervoltage, overcurrent, bus fault, and line fault. A power interruption happens when there is an insufficient amount of power being supplied to loads. This situation can occur during severe undervoltage conditions or when a path to a power source is absent. A local power interruption occurs when an individual load RPC is open. An upstream interruption occurs when a bus is isolated due to an upstream RPC being open. An overcurrent "soft" fault comes about when the component's filtered current reading is greater than the expected current

[1] The term synchronous is used synonymously for periodic.

value. A "hardware" overcurrent error occurs if the filtered current reading is greater than the hardware rating. When this situation arises the component is commanded off and an appropriate error message is sent to the operator. An undervoltage condition happens when all of the voltages across a bus are not within an acceptable tolerance of each other. Bus and line faults are determined by applying Kirchoff's Current Law. A fault transpires when the sum of all the currents across a bus or line are not within an acceptable tolerance. In the event that an anomaly in the system is detected, the operator is informed of the error. Additional information regarding the control system design is presented in [3].

Implementation of System Monitor

The functions of the power system monitor algorithm are distributed into several objects. Controlling the activation of the periodic events is the Synchronous_Manager object. The Synchronous_Manager prompts the 1553_Manager to read the sensor data. Additionally it causes each Sync_Proc to execute. Each individual component's Sync_Proc performs data conversion, filtering, and evaluation of power component health. The Saved_Data object acts as a repository for the filtered data. The System_Monitor object checks the components for anomalies, reports any errors detected to the PMC, performs bus and line fault detection, and once per second prompts Saved_Data to transmit the most recent snapshot of data to the PMC.

To accomplish the synchronous data collection, a timer was added to the control computers. An expansion card, with an 8354 programmable timer chip, was designed for the Compaq/386. The timer generates an interrupt every 100 milliseconds, at which time an Ada interrupt handler obtains control of the processor. The interrupt handler is a separate procedure of the Synchronous_Manager object, who can install and remove the handler. After receiving an interrupt, the handler activates a sequence of procedures to perform the data acquisition, filtering, error detection, and error reporting. Figure 3 illustrates this process and its interface to the rest of the testbed software.

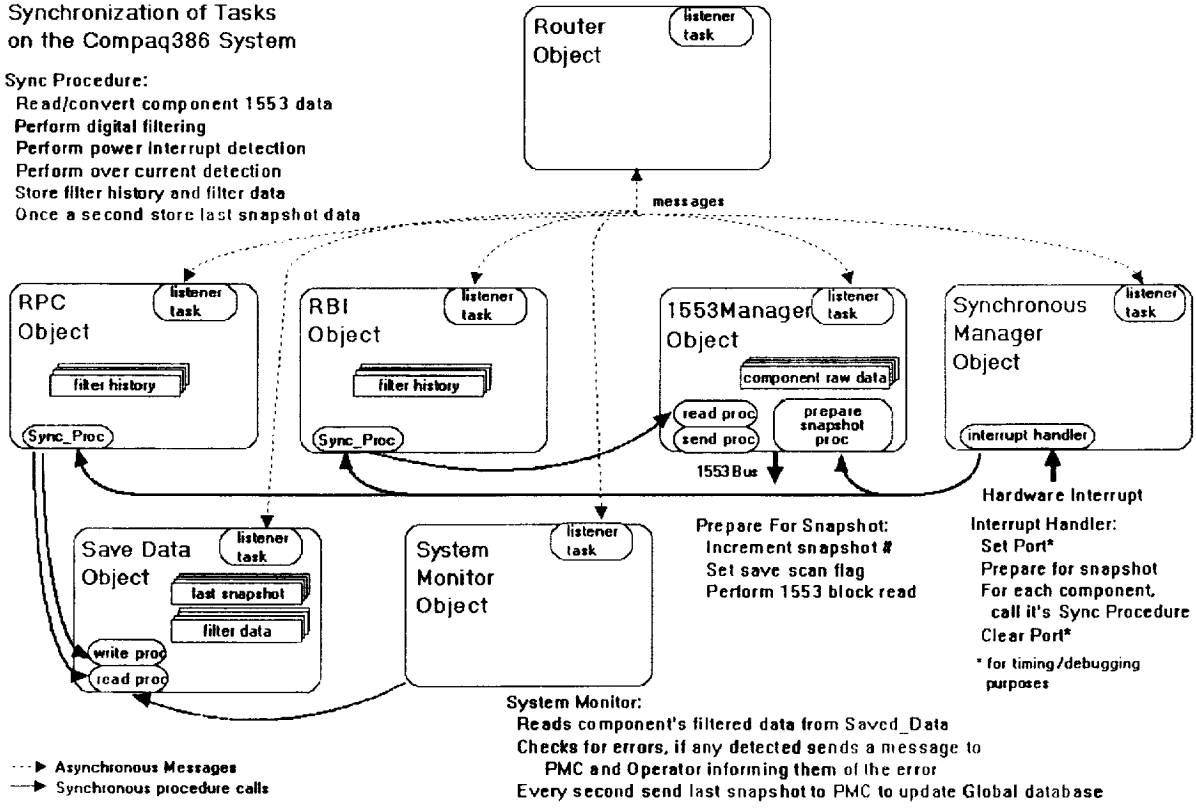

Figure 3 - Control of tasks for System Monitoring.

After the interrupt is received, the interrupt handler executes the procedure Prepare_for_Snapshot. This procedure reads the 1553 bus and obtains the sensor data from all components connected to it. Next, the interrupt handler calls the Sync_Proc procedure found in every power component object. Each Sync_Proc will get the raw data for its own particular component and convert it into floating point numbers. Next the Sync_Proc must retrieve the component's filter history and then use the Two_Hz_Filter procedure to smooth the data. Then it performs power interruption and overcurrent detection analysis. Once a second, it stores the filtered data into the last snapshot data array to be sent to the PMC. Before returning from the procedure, the filter data and filter history are stored for the next processing interval. After all of the component object's periodic procedures have been executed, the interrupt handler releases control of the processor to allow other Ada tasks to execute. When the interrupt handler releases control of the processor, the System_Monitor is the next task allowed to execute. The System_Monitor checks for any errors that need to be immediately reported to the PMC, and once per second sends the last snapshot to the PMC so a global database may be updated.

Assembly Routine for Procedure Activation

When executing the periodic activation of the Sync_Procs, the Ada rendezvous must be avoided because of the relatively long execution time necessary to perform it. Therefore, an assembly routine was written to bypass the Ada rendezvous. This routine is used to provide vector control from the interrupt handler to the periodic procedures. The assembly code directly jumps to the memory location of the Sync_Proc procedure, after setting up some local data on the stack. One complication in the assembly routine results from the component object being created at run time. Since the Sync_Proc code is defined as part of the code of each component object, its memory location is not known until run time. When a component's task is initialized by the testbed operator, the address of its Sync_Proc is stored in a table of active devices. This table resides within the Synchronous_Manager task and is accessed by the timer interrupt handler when appropriate.

Use of the assembly routine does in fact reduce execution time. A summary of the timing results are presented in the following section.

Timing Results

From the preliminary tests performed, it appears the implementation will be able to execute the system monitor functions within the 100 millisecond time frame. Compared to previous versions of the testbed software, execution times have been reduced by roughly ninety percent. The execution time of the assembly routine is 100 microseconds as opposed to the 3.5 milliseconds required to perform the Ada rendezvous. The overall execution time for performing the data acquisition, filtering, and fault detection has been reduced from 38 milliseconds to 4 milliseconds. The majority of time savings is due to bypassing the Ada rendezvous. In Figure 4, the different periodic operation's timings are shown. The vertical axis shows the periodic functions. The horizontal axis shows time.

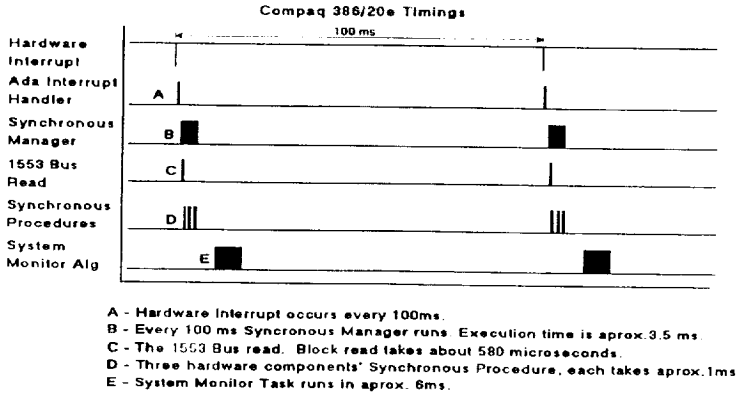

Figure 4 - Timings of the Synchronous Activities

Conclusion

The real-time system monitoring operations have been successfully implemented using Ada and the assembly language routine as described in this paper. By using the Ada language, a flexible and maintainable software environment has been provided. The timing constraints of the system monitoring functions have been met by utilizing the assembly language routine.

As the PMAD DC testbed is expanded to include the second power channel, the control software is being upgraded to incorporate objects to control these new components. Additionally, new control algorithms are being developed and incorporated into the system as they become available. These additions should have little effect on the design and performance of the System_Monitor.

Acknowledgements

The advanced development software team consists of Dave Gantose, Kim Ludwig, Mike Mackin, Jim Withrow, and Ted Wright. Tony Baez and Greg Kimnach produced the functional design of the system monitor and control operations.

References

[1] J. Soeder, R. Frye, and R.Phillips, "The Development of Testbeds to Support the Definition and Evolution of the Space Station Freedom Power System," *IECEC-91 Proceedings,*August, 1991.

[2] T. Wright, M. Mackin, and D. Gantose, "Development of Ada Language Control Software for the NASA Power Management and Distribution Testbed," *IECEC-89 Proceedings,* August, 1989.

[3] A. Baez and G. Kimnach, "Description of the Control System Design for the Space Station Freedom PMAD DC Testbed," *IECEC-91 Proceedings,* August, 1991.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM - 105157		

4. Title and Subtitle
Description of Real-Time Ada Software Implementation of a Power System Monitor for the Space Station Freedom PMAD DC Testbed

5. Report Date

6. Performing Organization Code

7. Author(s)
Kimberly Ludwig, Michael Mackin, and Theodore Wright

8. Performing Organization Report No.
E - 6444

10. Work Unit No.
474 - 42 - 10

9. Performing Organization Name and Address
National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135 - 3191

11. Contract or Grant No.

13. Type of Report and Period Covered
Technical Memorandum

12. Sponsoring Agency Name and Address
National Aeronautics and Space Administration
Washington, D.C. 20546 - 0001

14. Sponsoring Agency Code

15. Supplementary Notes
Prepared for the 26th Intersociety Energy Conversion Engineering Conference cosponsored by ANS, SAE, ACS, AIAA, ASME, IEEE, and AIChE, Boston, Massachusetts, August 4 - 9, 1991. Kimberly Ludwig, Sverdrup Technology, Inc., Lewis Research Center Group, 2001 Aerospace Parkway, Brook Park, Ohio 44142; Michael Mackin and Theodore Wright, NASA Lewis Research Center. Responsible person, Kimberly Ludwig, (216) 433 - 6251.

16. Abstract
This paper describes the Ada language software developed to perform the electrical power system monitoring functions for the NASA Lewis Research Center's Power Management and Distribution (PMAD) DC testbed. The results of the effort to implement this monitor will be presented. The PMAD DC testbed is a reduced-scale prototype of the electrical power system to be used in Space Station Freedom. The power is controlled by smart switches known as power control components (or switchgear). The power control components are currently coordinated by five Compaq 386/20e computers connected through an 802.4 local area network. One of these computers is designated as the control node with the other four acting as subsidiary controllers. The subsidiary controllers are connected to the power control components with a Mil-Std-1553 network. An operator interface is supplied by adding a sixth computer. The power system monitor algorithm is comprised of several functions including: periodic data acquisition, data smoothing, system performance analysis, and status reporting. Data is collected from the switchgear sensors every 100 milliseconds, then passed through a 2 Hz digital filter. System performance analysis includes power interruption and overcurrent detection. The reporting mechanism notifies an operator of any abnormalities in the system. Once per second, the system monitor provides data to the control node for further processing, such as state estimation. The system monitor required a hardware timer interrupt to activate the data acquisition function. The execution time of the code was optimized by using an assembly language routine. The routine allows direct vectoring of the processor to Ada language procedures that perform periodic control activities. A summary of the advantages and side effects of this technique will be discussed.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
ADA (programming language) Control Distributed processing	Unclassified - Unlimited Subject Category 62

19. Security Classif. (of the report)	20. Security Classif. (of this page)	21. No. of pages	22. Price*
Unclassified	Unclassified	8	A02

NASA FORM 1626 OCT 86

*For sale by the National Technical Information Service, Springfield, Virginia 22161

PRECEDING PAGE BLANK NOT FILMED